ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hlaváček** Jméno: **Radek** Osobní číslo: **393647**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav: **Katedra softwarového inženýrství**

Studijní program: **Informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh a implementace software pro řídící jednotku gramorádia

Název bakalářské práce anglicky:

The Radiogram Control Software Implementation

Pokyny pro vypracování:

Cílem práce je analýza, návrh a implementace řídícího a ovládacího SW pro řídící jednotku gramorádia Tesla tvořenou počítačem Raspberry PI 2B.
Gramorádio bude disponovat alespoň funkcemi přehrávání gramodesek, přehrávání rádií, nahrávání a zpětný poslech. Část funkcí bude možno ovládat pomocí ovládacích prvků gramorádia (tlačítka, otočné ovladače), část funkcí pouze přes ovládací software.
1) Ve spolupráci s vedoucím práce zhodnoťte technický stav gramorádia a nahraďte vadné či nevhodné komponenty.
2) Navrhněte a implementujte řídící server pro gramorádio na platformě Raspberry PI s MS Windows a REST API, který bude obsluhovat veškeré periferie gramorádia.
3) Zhodnoťte standardní uživatelské rozhraní gramorádia a vhodně namapujte podmnožinu funkcionalit řídícího serveru na tlačítka a otočné ovladače gramorádia.
4) Implementujte klientskou aplikaci umožňující demonstrovat všechny funkce gramorádia. Aplikace poběží ve webovém prohlížeči nebo na osobním počítači.

Seznam doporučené literatury:

Dodá vedoucí práce.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Bělohoubek, katedra číslicového návrhu FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **03.10.2016** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: _____

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra Softwarového inženýrství

Bakalářská práce

Návrh a implementace software pro řídící jednotku gramorádia

Radek Hlaváček

Vedoucí práce: Ing. Jan Bělohoubek

15. května 2017

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Ing. Janu Bělohoubkovi za pomoc s hardwarovou částí a trpělivost s mými nedostatky. Dále také rodině za finanční pomoc při studiu a přátelům, kteří na rozdíl ode mne věřili, že studium dokončím.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Hlaváček, Radek. *Návrh a implementace software pro řídící jednotku gramorádia*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práce se zabývá návrhem a implementací serveru běžícím na zařízení Raspberry Pi 2, který díky svému rozhraní umožňuje ovládat gramorádio. Aplikace je implementována pro Windows 10 IoT Core a obsahuje i klientskou část. Server je implementován v jazyce C# s knihovnou .NET Framework, klientskou aplikaci představuje webová aplikace implementována pomocí HTML a Javascriptu. Výsledkem práce je gramorádio, které se dá ovládat jak pomocí tlačítek, tak i pomocí webové klientské aplikace, např. z mobilního telefonu.

Klíčová slova Gramorádio, Raspberry Pi 2, Windows 10 IoT Core, C# server, webová klientská aplikace.

Abstract

This thesis is about design and implementation of the server running on Raspberry Pi 2, which can control radiogram. Application is implemented for Windows 10 IoT Core and even includes a client part. The server is implemented in C# with .NET Framework. The client application is a web application implemented with HTML and Javascript. The final product of this thesis is a radiogram that can be controlled by using the buttons on the radiogram or by the client web application.

Keywords Radiogram, Raspberry Pi 2, Windows 10 IoT Core, C# server, web client application.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V současné době existuje mnoho multifunkčních gramorádií v retro stylu. Bohužel tato gramorádia jsou většinou špatnou imitací starších gramorádií, mají spoustu zbytečných funkcí a naopak nemají mnou požadované vlastnosti. Je na čase se pokusit o změnu.

Podle mého názoru by gramorádio mělo vypadat jako Tesla ZZ IV a mít funkcionalitu moderního počítače. Ovládání klasicky pomocí tlačítek i pomocí jiného zařízení v síti by neměl být problém. Podobný produkt na českém trhu není, a proto se ve své práci budu věnovat přestavbě starého gramorádia, zejména ze softwarového hlediska.

Práce obsahuje návrh a implementaci software pro Raspberry Pi 2, který přemění gramorádio ze starého kusu nábytku na moderní zařízení. Rozhodl jsem se implementovat REST API a WebSocket server pro Windows 10 IoT Core, protože je to poměrně nová a neprozkoumaná platforma. Software je implementován pro příslušný hardware, proto práce obsahuje i krátký popis hardwarové části.

Kapitola 1

Cíl práce

Cílem práce je analýza, návrh a implementace řídícího a ovládacího softwaru pro řídící jednotku gramorádia Tesla tvořenou počítačem Raspberry Pi 2.

Gramorádio bude disponovat funkcemi přehrávání gramodesek, přehrávání rádií, nahrávání a zpětný poslech. Část funkcí bude možno ovládat pomocí ovládacích prvků gramorádia (tlačítka, otočné ovladače), část funkcí pouze přes klientskou aplikaci.

Kapitola 2

Analýza a návrh

2.1 Popis architektury

Má práce je postavená na architektuře klient–server. Aplikace s architekturou klient–server obsahuje jak server, tak i klientskou aplikaci. Komunikace mezi těmito komponentami je přes počítačovou síť.[7]

2.1.1 REST API

Měl jsem na výběr z několika konceptů pro výměnu zpráv přes síť pro ovládací rozhraní gramorádia. Hlavními kandidáty byli SOAP a REST.

SOAP posílá zprávy založené na XML a server ihned odpovídá na požadavky klienta. Jeho nevýhody jsou pomalé zpracování a zejména složitost.[13]

REST je defakto standard pro snadnou integraci systémů na různých programových prostředcích. Ve své implementaci potřebuje pouze TCP/IP a HTTP knihovny, které jsou dostupné na všech platformách. Zajišťuje schopnost systémů (počítačové systémy, internet) vzájemně si poskytovat služby a efektivně spolupracovat.[12] REST je podporován všemi známými internetovými prohlížeči, a proto jsem se ho rozhodl použít.

2.1.2 WebSocket

Na začátku mé práce jsem si myslel, že WebSockety nebudu potřebovat a vše udělám pomocí REST API. První myšlenka byla, že klient bude pouze jeden a po připojení se zeptá na aktuální stav a zbylé změny se zobrazí ihned lokálně bez potvrzení ze serveru. V mém návrhu toto řešení nedávalo smysl, takže jsem se rozhodl, že klientů bude neomezeně. Druhá myšlenka byla taková, že by se každý klient jednou za sekundu zeptal serveru na změny. Byla by to celkem spolehlivá metoda, ale při větším počtu klientů by to zbytečně zatěžovalo Raspberry Pi, i když by stav aplikace byl pořád stejný. Proto jsem se rozhodl pro nejelegantnější řešení, a to WebSockety.

Proč tedy nepoužívám pouze WebSockety, ale i REST API? Protože REST API podporuje více prohlížečů. V dnešní době podporuje WebSockety také většina známých prohlížečů, ale například v prohlížeči Opera Mini tato podpora chybí (viz tab. 2.1).

Prohlížeč	Podpora WebSocketu
IE	Ano
Edge	Ano
FireFox	Ano
Chrome	Ano
Safari	Ano
Opera	Ano
iOS Safari	Ano
Opera Mini	Ne
Android Browser	Ano
Chrome for Android	Ano

Tabulka 2.1: Podpora WebSocketů v prohlížečích [15]

2.1.3 Klientská aplikace

Měl jsem mnoho možností jak udělat klientskou aplikaci. První možnost byla nativní aplikace pro některý operační systém mobilních telefonů (iOS, Android, Windows 10 Mobile). Druhá byla aplikace pro Windows 10 nebo aplikace přímo na Raspberry Pi pro operační systém Windows 10 IoT Core a připojit k Raspberry monitor a nebo ji ovládat pomocí vzdálené plochy. Jako další se nabízela možnost udělat multiplatformní aplikaci pro mobilní telefony např. pomocí frameworku Ionic a Cordova. Já jsem však zvolil jinou možnost, a to udělat webovou aplikaci (viz obr. 3.6).

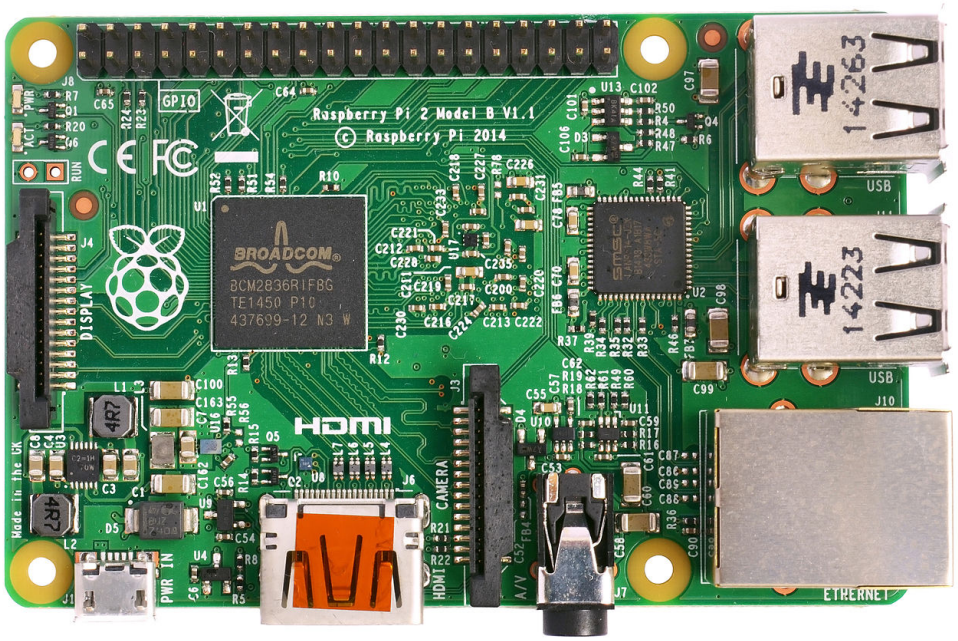
Webová aplikace může být používána jak z počítače s jakýmkoliv operačním systémem, tak i z mobilních telefonů. Jediná podmínka je být připojen do místní sítě a mít internetový prohlížeč. Aplikace posílá požadavky na REST API server a její stav se mění pomocí zpráv přijatých WebSocketem. Aplikace se při spuštění připojí na WebSocket server a pomocí REST požadavku si vyžádá aktuální stav, který je jí poslán zprávou WebSocketem. Při jakékoliv změně stavu se pošle zpráva všem připojeným klientským aplikacím. Tím pádem se na server může připojit více klientských aplikací a všechny vidí stejný stav.

2.2 Hardware

Po hardwarové stránce jsem spoléhal na svého vedoucího práce. Doporučil mi Raspberry Pi 2 nebo Arduino UNO. Po vyhledání podrobností o těchto

zařízeních jsem musel učinit rozhodnutí.

2.2.1 Raspberry Pi 2

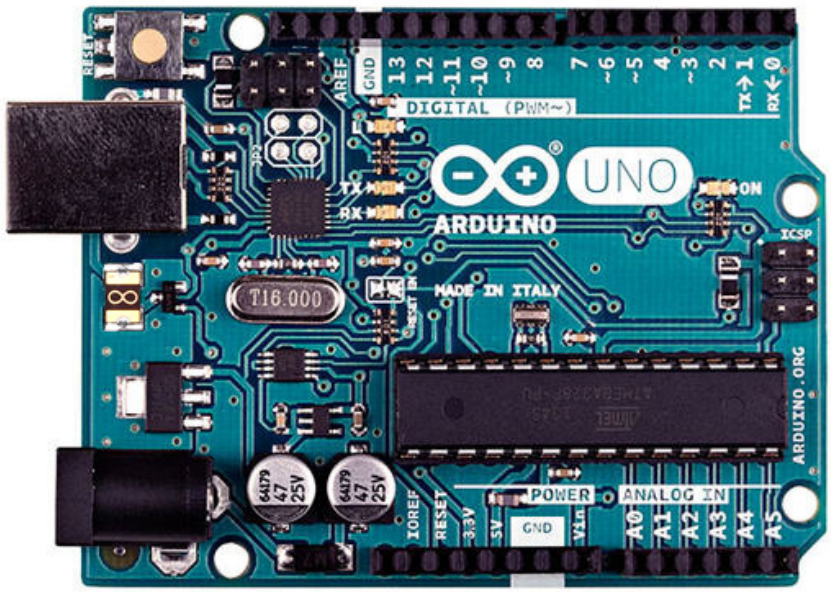


Obrázek 2.1: Raspberry Pi 2 [5]

Raspberry Pi 2 (viz obr. 2.1) je druhá generace Raspberry Pi, která má procesor ARMv7, a proto podporuje v plném rozsahu distribuce ARM GNU/Linux mezi které patří i Snappy Ubuntu Core a Microsoft Windows 10. Má 26 digitálních vstupů/výstupů.[11]

Ve své práci jsem se rozhodl použít Raspberry Pi 2 z důvodu kompatibility s Windows 10 a programovacím jazykem C#. Dalším faktorem byla neprozkoumanost schopností poměrně nového operačního systému Windows 10 IoT Core. Velkou výhodou je snadná práce s pamětí. Raspberry Pi má slot na paměťovou kartu, na které je nahrán operační systém a spousta volného místa.

2.2.2 Arduino UNO



Obrázek 2.2: Arduino UNO [2]

Arduino UNO (viz obr. 2.2) je mikrokontrolér s mikročipem ATmega328. Má čtrnáct digitálních vstupů/výstupů a 6 analogových vstupů/výstupů.[14] Práce se soubory není nemožná, ale je komplikovanější.

2.2.3 Raspberry Pi 2 vs Arduino UNO

Rozhodování mezi těmito zařízeními bylo celkem lehké. Jeden z důvodů pro Raspberry Pi byl můj obor studia. Většina práce s Arduinem nepatří do softwarového inženýrství. Další důvod byl snadná práce se soubory (viz tab. 2.2). Cílem bylo, aby moje aplikace byla schopná nahrávat a přehrávat audio a k tomu potřebuji velký úložný prostor s dobrým přístupem. Velká nevýhoda Raspberry Pi je, že nemá převodník analogového signálu na digitální. Raspberry Pi lze samozřejmě doplnit o tento převodník. Pro prototyp jsem zvolil jiný postup převodu analogového signálu na digitální a není úplně přesný. Pokud budu mít v budoucnu podobný problém, určitě Raspberry Pi o převodník doplním.

Vlastnost	Arduino UNO	Raspberry Pi 2
Analogový vstup	Ano	Ne
Multitasking	Ne	Ano
Jazyk C#	Ne	Ano
Open source	Ano	Ne
Operační systém	Ne	Ano
Snadné ukládání souborů	Ne	Ano

Tabulka 2.2: Porovnání Arduino UNO vs Raspberry Pi 2

2.3 Software

Podobný problém jsem zatím neměl a tudíž jsem vyhledával informace o různých variantách. Ihned jsem zjistil, že většina lidí, mající s Raspberry Pi zkušenosti, doporučuje Linux jako operační systém. Já jsem se však nenechal odradit od hledání i jiných možností. Nejvíce mne zaujala možnost pracovat s operačním systémem Windows 10 IoT Core.

2.3.1 Windows 10 IoT Core vs Linux

Po rozhodnutí, že budu pracovat s Raspberry Pi, mne čekalo další důležité rozhodnutí. Linux nebo Windows 10 IoT Core? Linux se používá častěji, je vyzkoušený a existuje velké množství dokumentace a návodů pro zprovoznění serveru a ovládání pinů. Vzhledem k časovému rozdílu uvedení obou systémů není na uvedené platformě – na rozdíl od GNU/Linuxu – Windows 10 IoT Core tak dobře etablován – ani z hlediska velikosti uživatelské základny, ani z hlediska množství dokumentace, dostupných knihoven, apod. Podporuje aplikace pro univerzální platformu Windows (UWP), tím pádem se dají využít knihovny .NET a jazyk C#. Existuje mnoho nástrojů pro operační systém Windows, které usnadňují vývoj a práci s Raspberry Pi. Mám zkušenosti s jazykem C# a rád objevuji nové věci.

Zvolil jsem si tedy operační systém Windows 10 IoT Core. Po krátkém výzkumu jsem zjistil, že existuje knihovna Windows Devices Gpio a já jsem byl nadšen. Jednoduché ovládání a čtení pinů. K tomu jsem ještě počítal s tím, že využiji nějakou implementaci Owinu[1] (např. Katana) a kontejner (např. Autofac). Bohužel při vývoji jsem zjistil, že implementace Owinu pro UWP aplikace neexistuje a veškeré kontejnery používají knihovny, které nejsou v UWP aplikaci podporovány. Při výzkumu jsem také narazil na tento problém: „Only a subset of pins are available to usermode. Most pins are reserved by the system and cannot be accessed from usermode“.[21] Tato věta říká, že pro uživatele mohou být nějaké piny Raspberry Pi nedostupné, protože jsou rezervované pro operační systém. Donutilo mne to otestovat všechny piny na mém Raspberry Pi následujícím kódem, abych zjistil, jestli budu mít potřebný počet volných pinů.

```
private static string TestGPIOs() {
  var message = "";
  for (int i = 0; i < 40; i++) {
    try {
      var pin = GpioController.GetDefault().OpenPin(i);
      message += "Gpio" + i.ToString("D2") + " OK" +
          Environment.NewLine;
    }
    catch (Exception e) {
```

[1] definice rozhraní mezi .NET webovým serverem a webovými aplikacemi

```
            message += "Gpio" + i.ToString("D2") + " " + e.Message +
                Environment.NewLine;
        }
    }
    return message;
}
```

Z výsledku jsem zjistil, které GPIO piny mohu použít a které ne (viz tab. 2.3).

GPIO pin	Result
00-01	Pin is not available. It is reserved by the system.
02-13	OK.
14-15	Pin is not available. It is reserved by the system.
16-27	OK.
28-34	Pin is not available. It is reserved by the system.
35	OK.
36-38	Pin is not available. It is reserved by the system.
39	Element not found.

Tabulka 2.3: Dostupnost GPIO pinů

2.3.2 REST API

REST API požadavky používají HTTP metody. Nejpoužívanější HTTP metody a jejich funkce jsou v následující tabulce 2.4.

HTTP metoda	Funkce
GET	Získává informace.
POST	Vytvoření zdroje.
PUT	Změna zdroje.
DELETE	Smazání zdroje.

Tabulka 2.4: Nejznámější HTTP metody

V mé aplikaci budu potřebovat čtyři druhy požadavků, čili budu potřebovat čtyři třídy (tzv. controllery). Budou to AudioController, RecordController, FMController a WebSocketController.

AudioController bude pouštět a vypínat audio vstup a přepínat mezi gramofonem a FM rádiem (viz tab. 2.5).

Metoda	HTTP metoda	Relativní URL	Funkce
Play	POST	/Audio	Přehraje audio vstup
Stop	DELETE	/Audio	Nepřehraje audio vstup
Switch	PUT	/Audio	Přepíná gramofon/rádio

Tabulka 2.5: Metody třídy AudioController

RecordController má za úkol nahrávání a přehrávání souborů. S tím souvisí i získání seznamu nahraných skladeb, možnost je smazat a i nastavit hlasitost přehrávání. Podrobnější seznam je v tabulce 2.6.

Metoda	HTTP metoda	Relativní URL	Funkce
StartRecording	GET	/Record	Nahraje vstup
StopRecording	PUT	/Record	Přestane nahrávat
Play	PUT	/Record/Play	Přehraje soubor
Pause	PUT	/Record/Pause	Přestane hrát
SetVolume	PUT	/Record/Volume	Nastaví hlasitost
Delete	DELETE	/Record	Vymaže soubor
Update	PUT	/Record/Update	Změní soubor
GetFile	GET	/Record/Download	Stáhne soubor
GetAll	GET	/Record/All	Seznam souborů

Tabulka 2.6: Metody třídy RecordController

FMController slouží k ovládání FM rádia. Nastavuje vyžádané frekvence, či spouští hledání stanic (viz tab. 2.7).

Metoda	HTTP metoda	Relativní URL	Funkce
SetFreq	POST	/FM	Nastaví frekvenci
Search	PUT	/FM	Hledá stanice
Mute	DELETE	/FM	Ztlumí rádio
CancelSearch	GET	/FM/Cance	Zastaví hledání

Tabulka 2.7: Metody třídy AudioController

WebSocketController na základě požadavku zkontroluje stav gramorádia, a pokud není nějaká část ve výchozím stavu, pošle zprávu klientským aplikacím s aktuálním stavem přes WebSocket (viz tab. 2.8).

Metoda	HTTP metoda	Relativní URL	Funkce
GetCurrentState	GET	/WebSocket	Pošle aktuální stav

Tabulka 2.8: Metody třídy AudioController

2.3.3 WebSocket

WebSocket je komunikační protokol poskytující obousměrnou komunikaci přes TCP.[17]

Každý klient se připojí na WebSocket server a server si uloží jeho spojení. Při jakékoliv změně server pošle zprávu pomocí WebSocketu všem uloženým spojením. Je to velmi jednoduché a elegantní řešení.

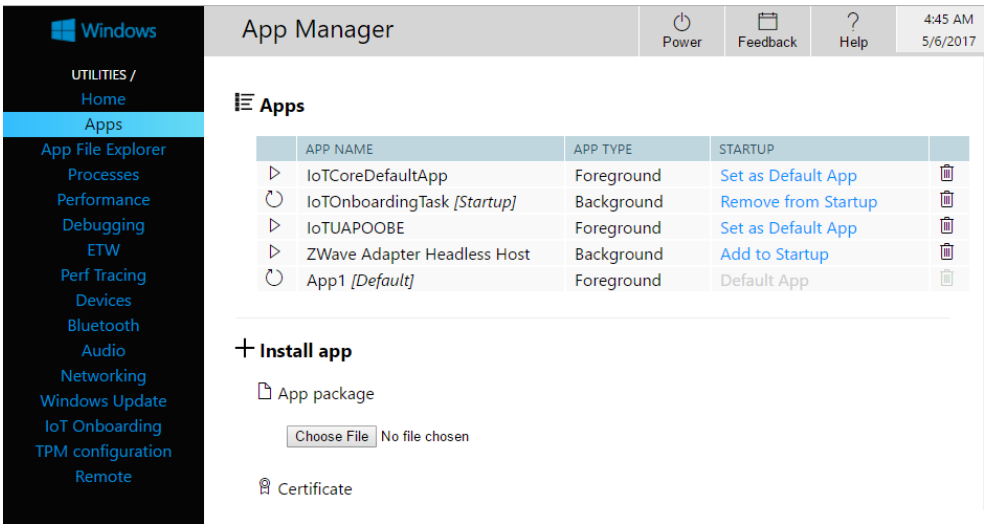
Kapitola 3

Realizace

3.1 Windows 10 IoT Core

Windows 10 IoT Core je verze operačního systému Windows 10, která je optimalizovaná pro menší zařízení např. pro Raspberry Pi 2 a 3. Tento operační systém využívá universální platformu Windows (UWP).[8] Microsoft nabízí také mnoho užitečných nástrojů pro vývoj a správu zařízení s tímto operačním systémem.

3.1.1 Dashboard

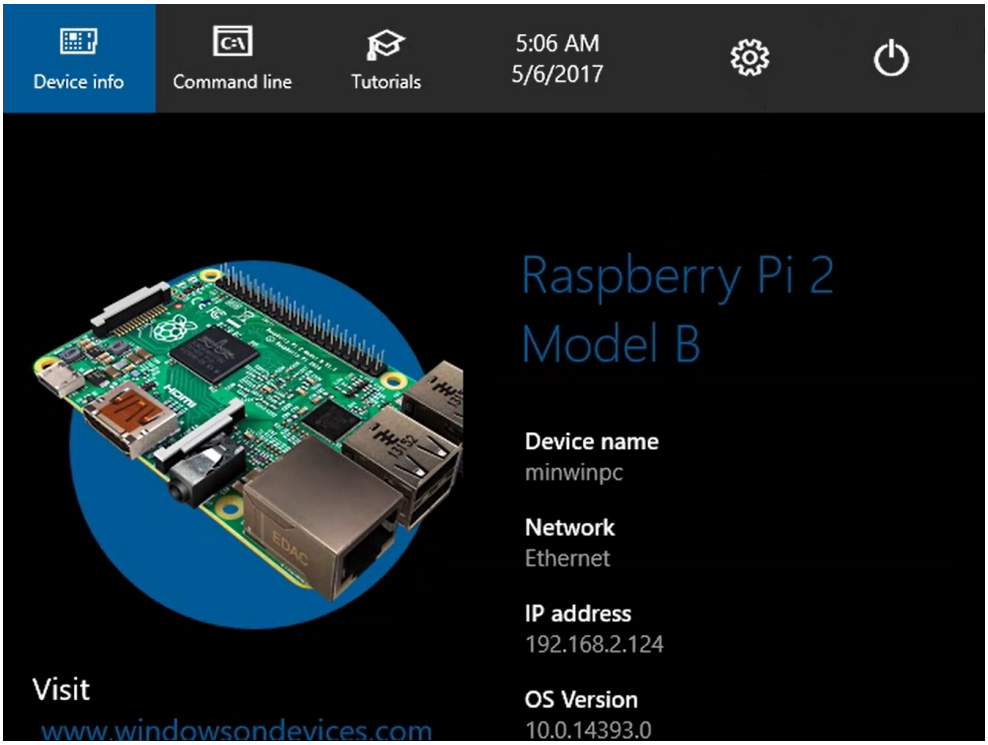


Obrázek 3.1: Portál zařízení

Windows 10 IoT Core Dashboard je desktopová aplikace pro správu zařízení jako Raspberry Pi 2. Aplikaci není součástí standardní instalace Windows 10,

takže ji musíme stáhnou ze stránek Microsoftu a nainstalovat. Pomocí této aplikace zjistíme typ zařízení, verzi operačního systému a především IP adresu. Na této adrese je přístupná moje klientská webová aplikace. Dále lze nastavit nové zařízení, nainstalovat Windows 10 IoT Core a otevřít portál zařízení (Device Portal, viz obr. 3.1), kde můžeme získat podrobný popis zařízení a můžeme spravovat například aplikace, procesy a soubory. Portál zařízení je webová aplikace na adrese zařízení na portu 8080.

3.1.2 Vzdálená plocha

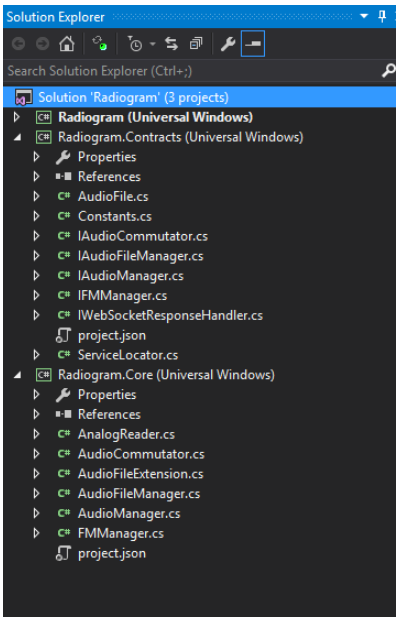


Obrázek 3.2: Windows 10 IoT Core zobrazen pomocí vzdálené plochy

Windows IoT Remote Client umožňuje připojení ke vzdálené ploše zařízení s nainstalovaným operačním systémem Windows 10 IoT Core (viz obr. 3.2). Aplikace je dostupná v Microsoft Store a na její stáhnutí a spuštění musíte být ve Windows 10 přihlášení pomocí Microsoft účtu. Aplikace je užitečná pro vývoj aplikací s uživatelským rozhraním přímo na zařízení nebo pro testování. V mé práci jsem vzdálenou plochu využil, ale jen pro oznámení, že moje klientská aplikace se tam nenachází, protože je to webová aplikace, a že je přístupná z lokální sítě.

3.1.3 Visual Studio

Visual Studio 2015 je vývojové prostředí, které jsem pro svoji práci využil. Podporuje mnoho programovacích jazyků a má hodně šablon projektů. Použil jsem šablonu pro Universal Windows projekt v jazyce C#.



Obrázek 3.3: Solution Explorer Visual Studia 2015

Když vytvoříme projekt ze šablony, Visual Studio vygeneruje solution (řešení) a základní projekt. V mém případě se jmenuje "Radiogram". Do solution můžeme vytvářet další projekty (knihovny), které teoreticky můžeme použít i v jiných projektech. Abychom v tom měli přehled, Visual Studio nabízí základní, ale velmi důležitou, komponentu a tou je Solution Explorer (průzkumník řešení, viz obr. 3.3).

Solution Explorer dokáže zobrazit a vytvářet projekty, přidávat reference na projekty, vytvářet složky, soubory atd. Pomáhá nám ve správném rozvržení projektů a usnadňuje orientaci v nich.

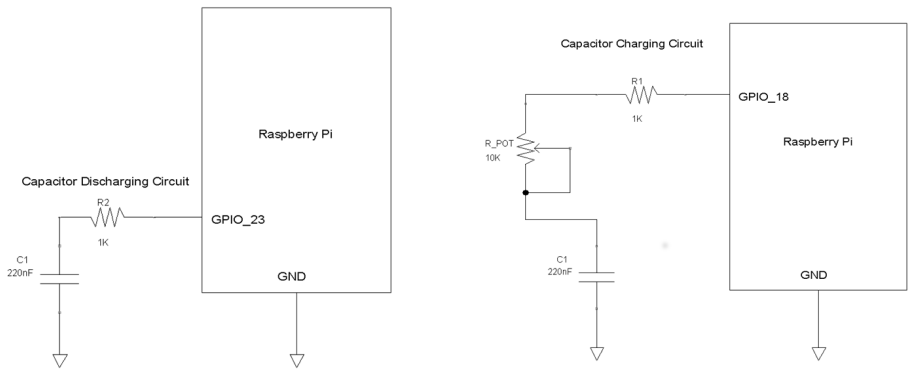
Visual Studio také nabízí šablony testovacích projektů. V mé práci jsem je bohužel nemohl využít, protože moje knihovny užívají knihovny specifické pro Raspberry Pi a mohou se používat pouze na daném zařízení.

3.2 Hardware

Podrobná analýza komponent hardwaru není součástí mé práce, protože tato práce se zabývá zejména softwarovou částí. Hardware jsem vybíral s vedoucím práce a jelikož hardwarová konfigurace úzce souvisí s mým softwarem, je zde stručně popsána.

3.2.1 Převod analogového signálu

Raspberry Pi 2 není schopno měřit analogový signál. Na vstup přijímá pouze digitální signál (0,1). Pro ladění frekvence FM rádia a nastavení hlasitosti v mé práci používám potenciometry, které mění odpor v obvodu. Za účelem měření odporu jsem vytvořil dvě smyčky, nabíjecí (viz obr. 3.4a) a vybíjecí (viz obr. 3.4b). Obě smyčky mají společný kondenzátor. Nabíjecí navíc obsahuje potenciometr, který ovládá hlasitost/frekvenci. Nejdříve kondenzátor vybiji pomocí vybíjecí smyčky bez přidaného odporu a poté měřím čas nabití kondenzátoru pomocí nabíjecí smyčky s potenciometrem (proměnným odporem). Pokud se kondenzátor nabije, na vstup do Raspberry Pi dostanu hodnotu 1. Čas nabití kondenzátoru s rostoucím odporem bude delší. Po zjištění nabíjecího času pro hraniční hodnoty potenciometru mohu po změření času nabíjení zjistit přibližnou hodnotu odporu potenciometru.[3] Tato metoda není moc přesná, ale pro účely ovládacích prvků gramorádia dostatečná.



(a) Vybíjecí smyčka (b) Nabíjecí smyčka

Obrázek 3.4: Nabíjecí a vybíjecí obvod kondenzátoru [3]

Výše uvedený postup musím pravidelně opakovat a zjišťuji, jestli se hodnota potenciometru nezměnila. Z tohoto důvodu jsem implementoval třídu AsyncActionRunner, která registruje akci, která se má v daném intervalu opakovat. Můžeme nastavit zpoždění prvního volání této akce např. z důvodu inicializace jiných komponent.

```
public class AsyncActionRunner : IDisposable {
```

```
  private List<DispatcherTimer> _timers = new
      List<DispatcherTimer>();

  public void ScheduleRepeatableAction(Action action, TimeSpan
      repeat, TimeSpan delay = default(TimeSpan)) {
    if (delay == default(TimeSpan)) delay = TimeSpan.FromSeconds(1);
    if (repeat == default(TimeSpan)) throw new
        ArgumentException("repeat");
    var timer = new DispatcherTimer();
    var first = true;
    timer.Interval = delay;
    timer.Tick += delegate {
      try {
        if (first) {
          timer.Interval = repeat;
          first = false;
        }
        action();
      }
      catch (Exception) { } //ignore error
    };
    _timers.Add(timer);
    timer.Start();
  }

  public void Dispose() {
    foreach (var timer in _timers) {
      timer.Stop();
    }
  }
}
```

3.2.2 FM rádio

Obrázek 3.5: FM rádio TEA5767 [10]

Ve své práci používám FM rádio TEA5767 (viz obr. 3.5), které komunikuje pomocí sběrnice I^2C. Rádio je především určené pro Arduino, ale lze použít s jakýmkoliv zařízením, které dokáže komunikovat sběrnicí I^2C.

I^2C je sběrnice, pomocí které můžeme komunikovat se zařízením pouze se dvěma obousměrnýma vodiči. Jeden vodič přenáší hodinový signál a druhý data.[6]

Adresa pro komunikaci FM rádia je 0x60 a přenášená data mají 5 bajtů.

3.2.3 Frekvence FM rádia

Frekvence FM rádia se nastaví pomocí sběrnice I^2C, přes kterou zašleme zprávu se slovem požadované frekvence. Slovo frekvence pro FM rádio je vypočítané tímto vzorcem:

$$N = \frac{4 \times (f_{RF} + f_{IF})}{f_{REF}} \tag{3.1}$$

N = hledaná decimální hodnota slova
f_{RF} = frekvence
f_{IF} = mezifrekvence = 225 kHz
f_{REF} = referenční frekvence = 32,768 kHz

Z tohoto vzorce si dokážeme odvodit vzorec pro výpočet aktuální frekvence z přijatého slova v decimální podobě:

$$f_{RF} = \frac{N \times f_{REF}}{4} - f_{IF} \tag{3.2}$$

f_{RF} = hledaná frekvence
N = decimální hodnota slova
f_{IF} = mezifrekvence = 225 kHz
f_{REF} = referenční frekvence = 32,768 kHz

Pro přečtení a zápis slova frekvence používáme 14 bitů a to bity 0-5 z prvního bajtu a všech 8 bitů z druhého bajtu.[10] FM rádio nenabízí pouze zápis a čtení frekvence, ale i jiná nastavení (viz tab. 3.1).

Režim	Bajt	Bit	Funkce
Čtení/zápis	0	0-5	První část slova frekvence
Čtení/zápis	1	0-7	Druhá část slova frekvence
Čtení	0	7	Značka připravenosti
Čtení	0	6	Překročení limitu přijímaných signálů
Čtení	2	7	Stereo
Zápis	0	6	Hledání
Zápis	0	7	Ztlumení
Zápis	2	1	Ztlumení levého kanálu
Zápis	2	2	Ztlumení pravého kanálu
Zápis	2	3	Vynucení mono
Zápis	2	5-6	Úroveň zastavení hledání
Zápis	2	7	Směr hledání
Zápis	3	1	Rušení stereo hluku
Zápis	3	3	Mírné ztlumení
Zápis	3	4	Frekvence hodinového signálu
Zápis	3	6	Pohotovostní režim

Tabulka 3.1: Nejdůležitější bity pro komunikaci s FM rádiem [10]

3.3 Software

Výhodou operačního systému Windows 10 IoT Core je dostupnost mnoha knihoven pro jazyk C#. Jazyk C# je jednoduchý a intuitivní. Zde je přehled nejdůležitějších balíčků či implementací.

3.3.1 NuGet

NuGet je správce balíčků pro platformy od Microsoftu (zahrnuje i .NET). Poskytuje možnost vytvořit či použít balíčky z centrálního úložiště.[19] Je integrován přímo do Visual Studia, proto je jeho používání velmi snadné.

3.3.2 Owin

Owin definuje rozhraní mezi .NET webovým serverem a webovými aplikacemi.[1] Existuje mnoho implementací tohoto rozhraní. Nejznámější implementací je Katana[18] od Microsoftu.

Katana zjednodušuje vývoj Owin webových aplikací na platformě .NET. I pomocí Owinu a Katany se Microsoft snaží o zjednodušení vývoje aplikací. Bohužel to neplatí pro UWP aplikace. Windows 10 IoT Core nepodporuje WCF a ani brzy nebude. Proto v době vývoje neexistovala implementace Owinu, která by používala knihovny dostupné pro tyto aplikace.

Musel jsem tedy najít jinou cestu, a to pomocí následujících NuGet balíčků.

3.3.3 Restup

Restup je NuGet balíček pro vytvoření statického i REST serveru.[16] Je vyvinut přímo pro Raspberry Pi 2 a snadno se používá. V průběhu testování jsem zjistil, že mi celá implementace nevyhovovala. Naštěstí tento balíček je implementován tak, aby se dal přizpůsobit mým potřebám. V tomto balíčku jsem musel implementovat svůj RouteHandler podle definovaného rozhraní, který třídí REST požadavky registrovaným třídám a jejich metodám. V ukázce je metoda, která vybírá správnou třídu a metodu z registrovaných tříd pomocí reflexe a vrací zpracovaný požadavek. Metoda dokáže volat asynchroní i synchroní metody registrovaných tříd a předává jim parametry získané z adresy požadavku.

```
public async Task<HttpServerResponse>
    HandleRequest(IHttpServerRequest request) {
  foreach (var controller in _controllers) {
    var methods = controller.Key.GetMethods().Where(m =>
        CheckIfCompatible(request, m));
    if (methods.Any()) {
      var obj = controller.Value;
      dynamic result;
      if (obj == null)
        obj = Activator.CreateInstance(controller.Key);
```

```
      var method = methods.First();
      var url = request.Uri.ToString();
      var i = url.IndexOf('?');
      var parameters = GetParameters(method.GetParameters(), i ==
          -1 ? "" : url.Substring(i + 1));
      ((ApiController)obj).Request = request;
      if (method.GetCustomAttribute<AsyncStateMachineAttribute>()
          != null)
        result = await ((dynamic)method.Invoke(obj, parameters));
      else
        result = method.Invoke(obj, parameters);
      ((ApiController)obj).Request = null;
      return result;
    }
  }
  return null;
}
```

Registrované třídy musí být potomci třídy ApiController, abych mohl přistupovat k požadavku na sever.

```
public class ApiController {
  public IHttpServerRequest Request;
}
```

Pro přehlednější hledání metod v těchto třídách používám atribut UriFormat z tohoto balíčku a vlastní atribut Method, kterým určuji jaký požadavek daná metoda zpracovává.

```
public sealed class MethodAttribute : Attribute {
  public MethodAttribute(HttpMethod method) {
    Method = method;
  }
  public HttpMethod Method { get; }
}
```

Použití mého formátu tříd je velmi jednoduché, přehledné a spolehlivé.

```
routeHandler.RegisterController<AudioController>();

public class AudioController : ApiController {
  [Method(HttpMethod.PUT)]
  [UriFormat("/Audio")]
  public async Task<HttpServerResponse> Switch() {
    await ServiceLocator.AudioCommutator.Switch();
    return
        MessageCreater.CreateResponseMessage(HttpResponseStatus.OK);
  }
}
```

Vzhledem k tomu, že Restup nepodporuje WebSockety, používám ještě další NuGet balíček, bez kterého bych se neobešel.

3.3.4 IoTWeb

IoTWeb je NuGet balíček, který umožňuje implementovat jednoduchý WebSocket server i v UWP aplikacích.[4] Pro správné chování jsem implementoval vlastní WebSocket handler. Je to jednoduchá implementace pro spravování spojení a posílání zpráv změn mým klientským aplikacím.

```
public class WebSocketHandler : IWebSocketRequestHandler,
    IWebSocketResponseHandler {
  private List<WebSocket> _connectedSockets = new List<WebSocket>();

  public void Connected(WebSocket socket) {
    lock (_connectedSockets) {
      _connectedSockets.Add(socket);
    }
    socket.ConnectionClosed += ConnectionClosed;
  }

  private void ConnectionClosed(WebSocket socket) {
    lock (_connectedSockets) {
      _connectedSockets.Remove(socket);
    }
  }

  public void SendMessageToConnectedWebSockets(Message message) {
    var stringMessage = JsonConvert.SerializeObject(message);
    lock (_connectedSockets) {
      foreach (var socket in _connectedSockets)
        socket.Send(stringMessage);
    }
  }
}
```

3.3.5 Objektový kontejner

Původně jsem měl v úmyslu použít kontejnery jako Unity nebo Autofac. Tyto kontejnery umožňují jednoduché registrování komponent a řeší závislosti těchto komponent na sobě. Tyto a jim podobné implementace kontejnerů používají knihovnu ASP.Net.MVC, která potřebuje IIS, což je webový server od Microsoftu.[20] Jelikož UWP aplikace, čili aplikace pro Raspberry Pi, nedokáží IIS spustit, nemohu tyto kontejnery používat. Měl jsem na výběr mezi implementací vlastního kontejneru a řešit závislosti jednotlivých komponent a nebo použít mnohem jednodušší řešení. Zvolil jsem jednodušší variantu a to vytvoření statické třídy ServiceLocator, do které při inicializaci mé aplikaci přiřadím jednotlivé komponenty.

```
public static class ServiceLocator {
  public static IFMManager FMManager { get; set; }
  public static IAudioCommutator AudioCommutator { get; set; }
  public static IAudioManager AudioManager { get; set; }
  public static IAudioFileManager AudioFileManager { get; set; }
```

```
  public static IWebSocketResponseHandler WebSocketResponseHandler
      {get; set;}
}
```

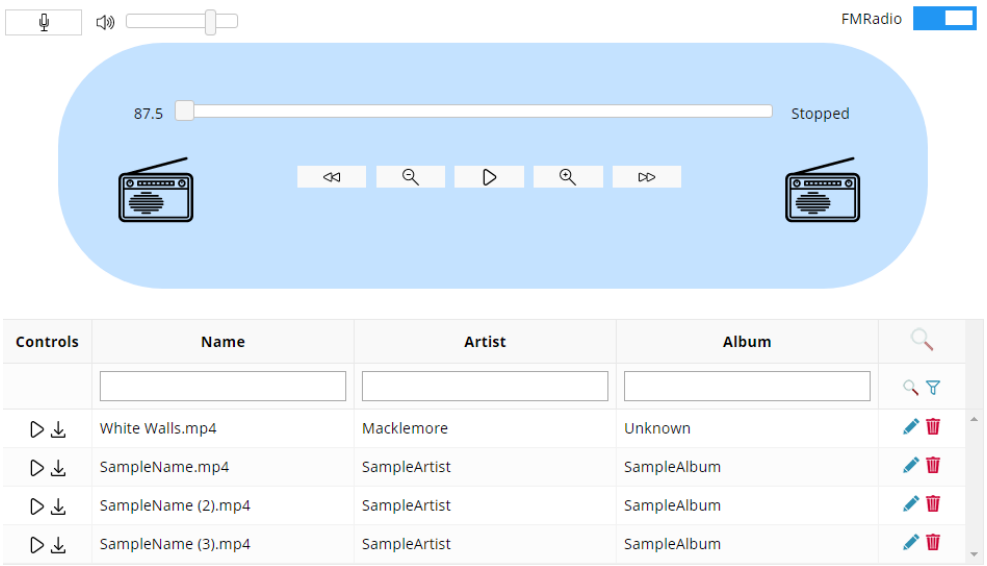
3.3.6 Audio manažer

Jeden z nejtěžších úkolů bylo naimplementovat nahrávání a přehrávání audio zvuku z gramofonu, rádia a i ze souboru. Potřeboval jsem naimplementovat třídu AudioManager tak, aby vše bez problému zvládala.

Nejprve jsem používal třídu MediaCapture z knihovny Windows Media Capture a třídu MediaPlayer z knihovny Windows Media Playback. Tyto dvě třídy dohromady teoreticky nabízely přesně to, co jsem potřeboval. Bohužel třída MediaPlayer nenabízela mixování vstupů a výstupů a proto by bylo velmi komplikované nahrávat a přitom přepínat audio vstupy. Tento přístup jsem musel opustit a použil jsem třídu AudioGraph z knihovny Windows Media Audio.

AudioGraph je dokonalá pro práci s audio zvukem, bohužel nemá dobře zpracovanou dokumentaci a příkladů užití také není moc. To mne ale nezastavilo. Výhoda jazyku C# je dobrá dokumentace pouze kódem a Visual Studio hodně pomáhá k orientaci v něm. AudioGraph umožňuje vytvořit mnoho vstupů a výstupů jak ze souboru, tak z gramofonu nebo rádia a můžu je jednotlivě zapínat a vypínat a popřípadě i mixovat.

3.3.7 Klientská aplikace



Obrázek 3.6: Webová klientská aplikace

Základ aplikace je HTML stránka (viz obr. 3.6). Její funkční vlastnosti jsou naimplementované v programovacím jazyku Javascript. Využívám knihovny jQuery a jsGrid, které jsem vybral na základě jejich dobré dokumentace, příkladů užití a ohlasů.

JsGrid je rychlá knihovna pro zobrazení dat do tabulky, která umožňuje rozsáhlé přizpůsobení každému uživateli. Kromě jiných funkcí nabízí filtrování, stránkování, řazení i editaci dat.[9] Můžeme nastavit události při změně dat, takže velmi ulehčuje udržení konzistence s uloženými daty na serveru (např. smazání nahrávky).

```
onItemDeleted: function (args) {
  $.ajax({
    type: "DELETE",
    url: "./api/Record",
    data: JSON.stringify(args.item),
    contentType: "application/json"
  });
}
```

Závěr

Výsledkem práce je prototyp gramorádia, které je možno ovládat pomocí klientské aplikace z webového prohlížeče. Prozkoumal jsem možnosti operačního systému Windows 10 IoT Core a povedlo se mi naimplementovat REST API a WebSocket server v UWP aplikaci.

S implementací jsem velmi spokojen, nicméně mohla by být jednodušší a přehlednější. Komplikace se objevily u implementace Owinu a implementace objektového kontejneru. Jelikož UWP aplikace nepodporují všechny knihovny jako jiné .NET aplikace, musel jsem se vyhnout ověřeným implementacím těchto komponent.

Nejsem s pokojen pouze s jedinou částí mé aplikace, která souvisela s hardwarem. Problém byl s chybějícím převodníkem analogového signálu na digitální u RaspBerry Pi 2, který by umožnil implementovat měření nastavení potenciometru elegantněji než pomocí měření času nabití kondenzátoru.

Celkově práci považuji za velký úspěch, prototyp gramorádia je funkční a ovládání jak pomocí tlačítek, tak pomocí klientské aplikace je spolehlivé.

Literatura

[1] About. *OWIN - Open Web Interface for .NET* [online] [cit. 7.5. 2017]. Dostupné z: http://owin.org/

[2] Arduino UNO. *Arduino* [online] [cit. 7.5. 2017]. Dostupné z: http://www.arduino.org/products/boards/arduino-uno

[3] Building Raspberry Pi Controllers Part 5: Reading Analog Data with an RPi. *All About Circuits* [online] [cit. 7.5. 2017]. Dostupné z: https://www.allaboutcircuits.com/projects/building-raspberry-pi-controllers-part-5-reading-analog-data-with-an-rpi/

[4] Embedded HTTP and WebSocket Server for UWP/.NET 4.5. *The world's leading software development platform* [online] [cit. 7.5. 2017]. Dostupné z: https://github.com/sensaura-public/iotweb

[5] Getting Started with the Raspberry Pi. *Raspberry Pi Foundation* [online] [cit. 7.5. 2017]. Dostupné z: https://www.raspberrypi.org/forums/viewtopic.php?t=4751

[6] I2C. *Wikipedie, otevřená encyclopedie* [online] [cit. 7.5. 2017]. Dostupné z: https://cs.wikipedia.org/wiki/I2C

[7] Klient-server. *Wikipedie, otevřená encyclopedie* [online] [cit. 12.5. 2017]. Dostupné z: https://cs.wikipedia.org/wiki/Klient-server

[8] Learn about Windows 10 IoT Core. *Windows Dev Center* [online] [cit. 7.5. 2017]. Dostupné z: https://developer.microsoft.com/en-us/windows/iot/Explore/IoTCore

[9] Lightweight Grid jQuery Plugin. *The world's leading software development platform* [online] [cit. 7.5. 2017]. Dostupné z: http://js-grid.com/

[10] Radio FM TEA5767. *Development Platform for Devices* [online] [cit. 7.5. 2017]. Dostupné z: https://developer.mbed.org/users/edodm85/notebook/radio-fm-tea5767/

[11] Raspberry Pi 2 Model B. *Raspberry Pi Foundation* [online] [cit. 7.5. 2017]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[12] Representational state transfer. *Wikipedia, the free encyclopedia* [online] [cit. 7.5. 2017]. Dostupné z: https://en.wikipedia.org/wiki/Representational_state_transfer

[13] SOAP. *Wikipedie, otevřená encyclopedie* [online] [cit. 12.5. 2017]. Dostupné z: https://cs.wikipedia.org/wiki/SOAP

[14] Start with Arduino UNO. *Arduino* [online] [cit. 7.5. 2017]. Dostupné z: http://www.arduino.org/learning/getting-started/getting-started-arduino-uno

[15] Web Sockets. *Can I use...* [online] [cit. 7.5. 2017]. Dostupné z: http://caniuse.com/websockets

[16] Webserver for Universal Windows Apps. *The world's leading software development platform* [online] [cit. 7.5. 2017]. Dostupné z: https://github.com/tomkuijsten/restup

[17] WebSocket. *Wikipedia, the free encyclopedia* [online] [cit. 7.5. 2017]. Dostupné z: https://en.wikipedia.org/wiki/WebSocket

[18] Welcome to Katana. *The world's leading software development platform* [online] [cit. 7.5. 2017]. Dostupné z: https://github.com/aspnet/AspNetKatana/

[19] What is NuGet? *NuGet Gallery* [online] [cit. 7.5. 2017]. Dostupné z: https://www.nuget.org/

[20] B., P.: Use Microsoft.AspNet.Mvc in Universal Windows Platform (UWP) Application. *Stack Owerflow* [online]. 2016, 24.2. 2016 [cit. 7.5. 2017]. Dostupné z: http://stackoverflow.com/questions/35598648/use-microsoft-aspnet-mvc-in-universal-windows-platform-uwp-application

[21] Spieker, S.: Failed to open a handle to the device when opening GPIO pin. *Stack Owerflow* [online]. 2015, 27.10. 2015 [cit. 7.5. 2017]. Dostupné z: http://stackoverflow.com/questions/30579701/failed-to-open-a-handle-to-the-device-when-opening-gpio-pin

Příloha **A**

Seznam použitých zkratek

HTML HyperText Markup Language

REST Representational state transfer

SOAP Simple Object Access Protocol

XML Extensible Markup Language

API Application programming interface

ARM Advanced RISC Machine

GNU GNU is Not Unix

UWP Universal Windows Platform

GPIO General–purpose input/output

TCP Transmission Control Protocol

IP Internet Protocol

FM Frequency modulation

I^2C Inter–Integrated Circuit

WCF Windows Communication Foundation

MVC Model–view–controller

IIS Internet Information Services

Příloha **B**

Obsah přiloženého CD

```
readme.txt ..... stručný popis obsahu CD
exe ..... adresář se spustitelnou formou implementace
src
    impl ..... zdrojové kódy implementace
    thesis ..... zdrojová forma práce ve formátu LaTeX
text ..... text práce
    thesis.pdf ..... text práce ve formátu PDF
    thesis.ps ..... text práce ve formátu PS
```